```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
data = pd.read_excel('cancer.xlsx')
data.head()
```

index	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
1	842302	17.99	10.38	122.80	1001.0	0.1184
2	842517	20.57	17.77	132.90	1326.0	0.0847
3	84300903	19.69	21.25	130.00	1203.0	0.1096
4	84348301	11.42	20.38	77.58	386.1	0.1425
5	84358402	20.29	14.34	135.10	1297.0	0.1003

ows × 36 columns

```
# Prepare the model
y = data["diagnosis"] # our target variable
X = data.drop(["diagnosis","index","id"], axis=1) # our predictors
X.shape
```

```
    (569, 33)
```

```
# Taking care missing data
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(X.iloc[:, 0:29])
X.iloc[:, 0:29] = imputer.transform(X.iloc[:, 0:29])
```

```
# One hot encoding
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [30,31,32])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

```
# Spliting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
#Feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_train
```

```
    array([[-1.49240501,  2.21735578, -0.40488817, ..., -0.38664354,
             0.32349851, -0.7578486 ],
           [ 0.67005939, -0.45098762, -0.40488817, ..., -1.48895322,
             0.62563098, -1.03071387],
            [-1.49240501,  2.21735578, -0.40488817, ...,  0.71907312,
             -0.51329768, -0.96601386],
            ...,
            [ 0.67005939, -0.45098762, -0.40488817, ..., -1.01972052,
             -0.69995543, -0.12266325],
            [ 0.67005939, -0.45098762, -0.40488817, ..., -1.80208475,
             -1.56206114, -1.00989735],
            [-1.49240501,  2.21735578, -0.40488817, ..., -0.30719919,
             -1.24094654,  0.2126516 ]])
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
y_pred = classifier.predict(X_test)
y_pred_array = np.array(y_pred)
y_test_array = np.array(y_test)

# Reshape arrays
y_pred_reshaped = y_pred_array.reshape(len(y_pred_array), 1)
y_test_reshaped = y_test_array.reshape(len(y_test_array), 1)

# Concatenate arrays along the second axis
concatenated_array = np.concatenate((y_pred_reshaped, y_test_reshaped), axis=1)

print(concatenated_array)
```

```
 ['B' 'B']
 ['M' 'M']
 ['B' 'B']
 ['M' 'M']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['M' 'M']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['B' 'M']
 ['M' 'M']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['M' 'M']]
```

```
# Confusion metrics
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)*100
```

```
[[89  1]
 [ 7 46]]
94.4055944055944
```